Lab 6, Jupyter and Me

Due date: Wednesday, May 1, 10:00am

Note: This is a *very short* assignment designed to give you some basic familiarity with the work of Jupyter and with Jupyter notebooks. We are starting a new topic on May 1, so your work shall be completed by then.

Lab Assignment

Assignment Preparation

This is an individual lab.

The lab will be completed using Jupyter. To prepare, perform the following actions:

1. Access your Jupyter account on http://data301b.calpolydatascience.org server. Start your Jupyter server.
2. Open linux terminal.
3. Change to your assignments directory (create one if you have not done so).
4. In your assignments directory, please run the following command:

```
$ nbgrader fetch class=dekhtyar Lab6
```

As a result of the last command, a directory named `Lab6` will appear in your assignments directory. This directory will contain one iPython notebook: `BasicVectorOps.ipynb`.

You are now ready for the assignment. The assignment consists of two parts:

1. Coding Basic Vector Operations. For this part, you will complete the assignment in the `BasicVectorOps.ipynb` notebook.

2. Building Your Own Notebook. Here, you will build your own notebook for analyzing the Baby Names dataset that you have first encountered in **Lab 1**.

Part 1: Vector Operations

Complete the assignment in the `BasicVectorOps.ipynb` notebook. The assignment asks you to implement a collection of vector operations, using Python lists to represent the vectors.

For each function, its definition is given in a `markdown cell` and a `Python3` cell with a function header is provided to you. Complete the code in the python cell, make sure your function passes automated tests and - if necessary, test it further (you can add cells to the notebook).

Part 2: Analysis of Baby Names

I have placed the `NationalNames.csv` file containing the entirety of the Kaggle's *Baby Names* dataset in the ∼`dekhtyar/data/` directory. To determine if you can access it, try

```
$ ls -l ~dekhtyar/data
```

command. If you see the following response:

```
dekhtyar@data301b:~$ ls -l ~dekhtyar/data/
total 43312
-rw-r--r-- 1 dekhtyar dekhtyar 44350518 Apr 25 04:27 NationalNames.csv
```

you can access the data. **Use the file provided. Do NOT make extra copies of it.**

Recall, that `NationalNames.csv` file has the following format:

```
Id, Name, Year, Gender, Count
```

where, `Id` is the id of the row, `Name` is a specific name, `Year` is a year for which this name is considered (from 1880 to 2014), `Gender` is either `"M"` or `"F"` signifying whether the information is about a birth name given to boys or to girls, and `Count` is the number of times the name was assigned to a baby of a given birth gender in a given year.

The file contains a total of over 44 million records. Jupyter notebooks can accommodate that number of records in main memory, so you shall simply read all the data in and store it in data structure(s) you deem appropriate (we will discuss some possible data structures in class on April 25).

Managing this data may require some care. Make sure that your operations do not use multiple passes over the data structures where one pass is sufficient. This may mean a difference between a 3-second run and a 3-hour run for your code. A single pass of all the data is actually not too slow - you can upload all the data into main memory from a file in a matter of just a few seconds.

Your task. You will create a Jupyter notebook `names.ipynb` for analyzing the baby names dataset. The specific things to put into the notebook are listed below.

Notebook structure. Your notebook shall contain all the code required for analysis, all the code that actually runs the analysis and produces output. It shall also contain `markdown cells` with explanations and notes.

The general structure of the notebook is:

1. Header Cell(s). Start your notebook with one or more (whatever looks aesthetically better) `markdown` header cells. The header cell shall contain the title of the notebook, the course information, the lab number and your name. *In general,* all notebooks you create shall always contain a header cell (or cells) that contain this information.

2. Data Import. Put in one or more Python cells to import the baby names data into the notebook, place it in appropriate data structures (which you can treat as global variables for the rest of the notebook), and confirm that the data structures are built. **Do not output the full list of names**, but make sure to spot-check the data structure(s) you created.

3. Data analysis. For each analytical question asked below, include a set of cells that contains code and analysis of the results in response to the question. Each such section shall have the following format:

   (a) Task Header cell. Use one or more `markdown cells` to put a task header, and provide a quick overview of the analytical question that is being asked.

   (b) Solution cells. Provide one or more Python cells, intermixed with `markdown` cells as needed (for explanations and such) which contain the code you wrote to answer the analytical question, test your code, and provide the actual answer. If these cells are run one after another, the last code cell or cells shall generate the output requested by the analytical question.

   (c) Summary/analysis cell(s). Finish each task by including a `markdown` cell or cells containing analysis and discussion of the observed results. What did you find? Is it an interesting observation?

4. Final Summary. Include one or more `markdown cells` at the end of the notebook summarizing your experience with the data analysis of the

baby names. What have you learned? What interesting information have you learned? What other new questions related to the dataset would you like to answer?

List of tasks. You are asked to answer the following analytical questions.

1. **Question 1.** Given a name (and a gender), find a way to determine when this name was most popular. This is broken down into a number of subquestions:

   - First and foremost, given a name/gender pair, write a function `bestYear(name, gender)` that returns a tuple containing a list of years when the name reached its highest popularity (in terms of total number of names given), and the total number of names given in each of those years. For example if a male name `'Rostislav'`[1] appeared 100 times in years 2010 and 2014 and it was the largest number of times it was given, `bestYear(X, 'M')` shall return

     `([2010, 2014], 100)`

   - Second, write a function `bestYearRank(name,gender)`, which returns information in the same format as `bestYear()`, except rather than returning the raw number of times a name was given to a baby in the best year, it returns the years when the name achieved the peak of its popularity. E.g., if female name `'Polly'` reached the rank of 224 in years 1920 and 1927 and it was the highest rank achieved for this name [2] the output of `bestYearRank('Polly', 'F')` shall be

     `([1920, 1927], 224)`

     Note that the actual counts of the name `"Polly"` in those two years may be different.

   - Additionally, you shall explore the ups and downs in the total count and the rank of a name and develop two more functions `localPeaks(name, gender)` and `localPeaksRank(name,gender)` that given a name and a gender study the changes in the total number of times a name is given and in the rank of the name over time and report all local maxima for the respective lists that match the following conditions:
     - a value is a *local maximum* of a series (list) of numbers, if it is greater (or greater or equal) than both the preceding and the following value.
     - a value shall be returned by the `localPeaks(name, gender)` or `localPeaksRank(name, gender)` if it is a local peak, and there are no other local peaks higher than it within the 11

[1] I do not know if this is the case.
[2] I actually have no idea if this is the case.

year span around the year to which this value belongs (i.e., it is the highest local peak in the range of *current year - 5* to *current year +5* years).

In addition to `localPeaks()` and `localPeaksRank()` functions, develop `localCanyons()` and `localCanyonsRank()` functions that instead of reporting the highest values, report the lowest values (local minima) using the same set of rules.

All four functions shall output a list of tuples `(year, value)`, where the `year` is the year of the local optimum reported, and the value is the total count or the rank (depending on the function) of the name in that year.

Select a collection of names that are of interest to you (your name, names of your parents, siblings, friends, other names that might interest you), and analyze the popularity of the names in your collection using the tools you developed. Report on your results.

2. Question 2. Find the most popular names over a period of time. Given a range of years, and a gender, your goal is to determine which names were the most popular during that time, and to return the appropriate information.

   There are two good metrics of a name popularity:

   - **Total number of babies** who were given the name during the specified range of years.
   - **Average annual rank** of the name during the specified range of years.

   Write functions that given a range of years and a gender, compute these two metrics for the names found in the given range of years, and output the requested number of top names based on each metric. For example, you could name a function that uses the first metric `findPopularByCount()` and give it four parameters:

   ```
   def findPopularByCount(gender, startYear, endYear, k):
   ```

   where the $[startYear, endYear]$ range is **inclusive**, and `k` is the number of names to return. The output shall contain the name, and for each name, the value of the computed metric.

   Additionally, implement two functions `popularityByCount(name, gender, startYear, endYear)` and `popularityByRank(name, gender, startYear, endYear)` which find the popularity metrics for a given range of years for a given name-gender pair.

   Using these tools, compare the popularity of the set of names you selected (see Question 1) to the most popular male and female names in different time periods. (Select interesting and meaningful time periods, e.g., the Great Depression time, the World War II time, specific presidencies, or other time periods that make sense to you). Report and discuss your findings.

3. **Question 3.** Your question goes here. Determine what information you would like to find out that the dataset of baby names can provide for you. The specific question you ask has to be **new** - i.e., it cannot be one of the questions you studied in **Lab 1**, nor can it be any of the questions covered in class on April 25. You can **use** any of the code from **Lab 1** and from the April 25 notebooks, as well as any code you developed to answer Questions 1 and 2 to help answer your Question 3, but:

   - your question should be about the same level of difficulty (and no significantly less difficult then) Questions 1 and 2;
   - answering your question shall involve writing at least one (and possibly more) new function.

   For the question you are asking, write all functions that would help you analyze the data appropriately, and then run these functions on the inputs that are appropriate for the question. If appropriate, you an use the list of names from Questions 1 and 2. Discuss your findings.

Submission

For this lab we will use the Jupyter's `nbgrader` tool for submission.

You will be submitting the `Lab 6` directory that you have fetched using `nbgrader` earlier. The directory shall contain two Jupyter notebooks, `BasicVectorOps.ipynb` and `names.ipynb`

Use the following submission command:

```
$ nbgrader submit --course=dekhtyar Lab6
```

If things work as they are designed to work, you will successfully submit both notebooks.

Good Luck!